HITextView Unleashed on Tiger

Session 111

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

HITextView Unleashed on Tiger

Daniel Fenwick
Engineering Manager

http://developer.apple.com/wwdc2005/

Target Audience

- Carbon developers who need editable, scrollable text fields in their applications
- Developers currently using MLTE who want more ease of use
- Developers interested in new MLTE features in Mac OS X 10.4 Tiger

What You'll Learn

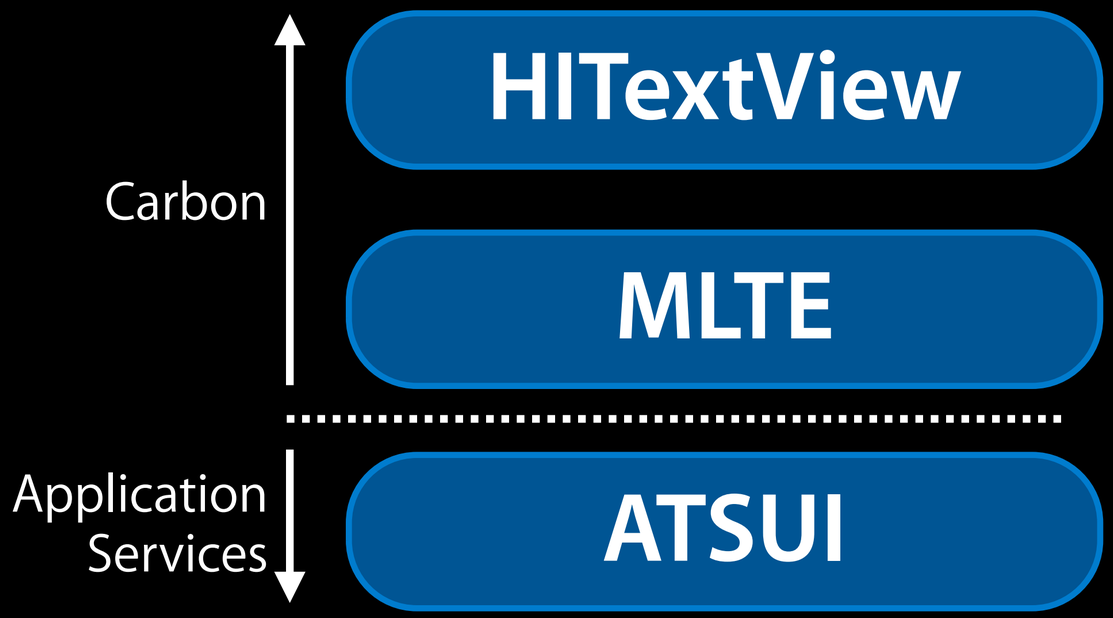
- Instantiating HITextView in your application
  - Interface Builder
  - Programmatically
- Embedding HITextView in other views
  - Compositing
  - Scrolling
  - Resizing
- Enabling UI features
  - Font and spelling panel support
  - Font substitution
- Getting data into and out of HITextView

What You'll Learn

- Instantiating HITextView in your application
  - Interface Builder
  - Programmatically
- Embedding HITextView in other views
  - Compositing
  - Scrolling
  - Resizing
- Enabling UI features
  - Font and spelling panel support

  - Font substitution
- Getting data into and out of HITextView

Framework Hierarchy



- HITextView provides highest level of abstraction
- Objects at lower levels can be accessed for greater control
- Some MLTE APIs are disabled when using HITextView

HITextView vs. MLTE

- Advantages
  - Better Interface Builder integration
  - HIView ordering and compositing
    - Scrolling and resizing
    - Transparency
  - More automatic event handling
    - Menu updating
    - Focus
  - Less code to write!
- Disadvantages
  - Some MLTE features are disabled

Let’s Do Some Coding!

http://developer.apple.com/wwdc2005/

Getting Started With HITextView

- Single call to create
- MLTE-specific options can be passed at creation time

```
HIViewRef textView;
HIRect bounds;
TXNFrameOptions options;

// ...

status = HITextViewCreate(&bounds, 0, options, &textView);
```

Getting Started With HITextView

- Single call to create
- MLTE-specific options can be passed at creation time

```
HIViewRef textView;
HIRect bounds;
TXNFrameOptions options;

// ...


status = HITextViewCreate(&bounds, 0, options, &textView);

// 1st parameter: Bounds of the view, set however you prefer
```

Getting Started With HITextView

- Single call to create
- MLTE-specific options can be passed at creation time

```
HIViewRef textView;
HIRect bounds;
TXNFrameOptions options;

// ...


status = HITextViewCreate(&bounds, 0, options, &textView);

// 1st parameter: Bounds of the view, set however you prefer
// 2nd parameter: HITextView-specific options, reserved for future use
```

Getting Started With HITextView

- Single call to create
- MLTE-specific options can be passed at creation time

```
HIViewRef textView;
HIRect bounds;
TXNFrameOptions options;

// ...


status = HITextViewCreate(&bounds, 0, options, &textView);

// 1st parameter: Bounds of the view, set however you prefer
// 2nd parameter: HITextView-specific options, reserved for future use
// 3rd parameter: MLTE-specific options
```

Getting Started With HITextView

- Single call to create
- MLTE-specific options can be passed at creation time

```
HIViewRef textView;
HIRect bounds;
TXNFrameOptions options;

// ...

options = kTXNDoFontSubstitutionMask;
status = HITextViewCreate(&bounds, 0, options, &textView);

// 1st parameter: Bounds of the view, set however you prefer
// 2nd parameter: HITextView-specific options, reserved for future use
// 3rd parameter: MLTE-specific options
```

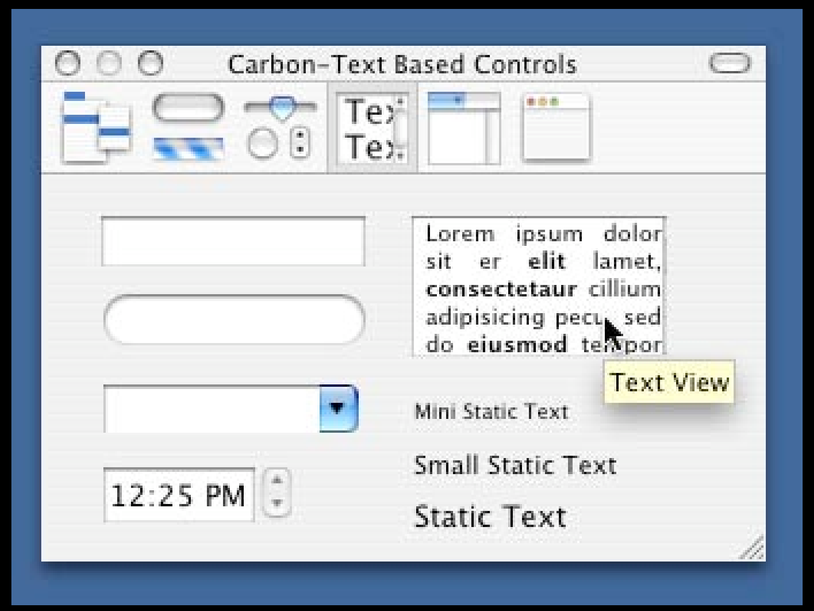
Getting Started With HITextView

- Single call to create
- MLTE-specific options can be passed at creation time

```
HIViewRef textView;
HIRect bounds;
TXNFrameOptions options;

// ...

options = kTXNDoFontSubstitutionMask;
status = HITextViewCreate(&bounds, 0, options, &textView);

// 1st parameter: Bounds of the view, set however you prefer
// 2nd parameter: HITextView-specific options, reserved for future use
// 3rd parameter: MLTE-specific options
// 4th parameter: The text view object
```

Getting Started With HITextView

Can also be created in Interface Builder



Getting Started With HITextView

Adding resize support

```
HIViewRef textView;
HILayoutInfo layoutInfo;

// ...

layoutInfo.binding.left.kind = kHILayoutBindLeft;
layoutInfo.binding.left.toView = NULL;
layoutInfo.binding.left.offset = 0.0;

// repeat for top, bottom, and right edges

status = HIViewSetLayoutInfo(textView, &layoutInfo);
```

Getting Started With HITextView

Adding resize support

```
HIViewRef textView;
HILayoutInfo layoutInfo;

// ...

layoutInfo.binding.left.kind = kHILayoutBindLeft;
layoutInfo.binding.left.toView = NULL;
layoutInfo.binding.left.offset = 0.0;

// repeat for top, bottom, and right edges

status = HIViewSetLayoutInfo(textView, &layoutInfo);
```

Getting Started With HITextView

Adding resize support

```
HIViewRef textView;
HILayoutInfo layoutInfo;

// ...

layoutInfo.binding.left.kind = kHILayoutBindLeft;
layoutInfo.binding.left.toView = NULL;
layoutInfo.binding.left.offset = 0.0;

// repeat for top, bottom, and right edges

status = HIViewSetLayoutInfo(textView, &layoutInfo);
```

Getting Started With HITextView

Adding resize support

```
HIViewRef textView;
HILayoutInfo layoutInfo;

// ...

layoutInfo.binding.left.kind = kHILayoutBindLeft;
layoutInfo.binding.left.toView = NULL;
layoutInfo.binding.left.offset = 0.0;

// repeat for top, bottom, and right edges

status = HIViewSetLayoutInfo(textView, &layoutInfo);
```

Getting Started With HITextView

Adding resize support

```
HIViewRef textView;
HILayoutInfo layoutInfo;

// ...

layoutInfo.binding.left.kind = kHILayoutBindLeft;
layoutInfo.binding.left.toView = NULL;
layoutInfo.binding.left.offset = 0.0;

// repeat for top, bottom, and right edges

status = HIViewSetLayoutInfo(textView, &layoutInfo);
```

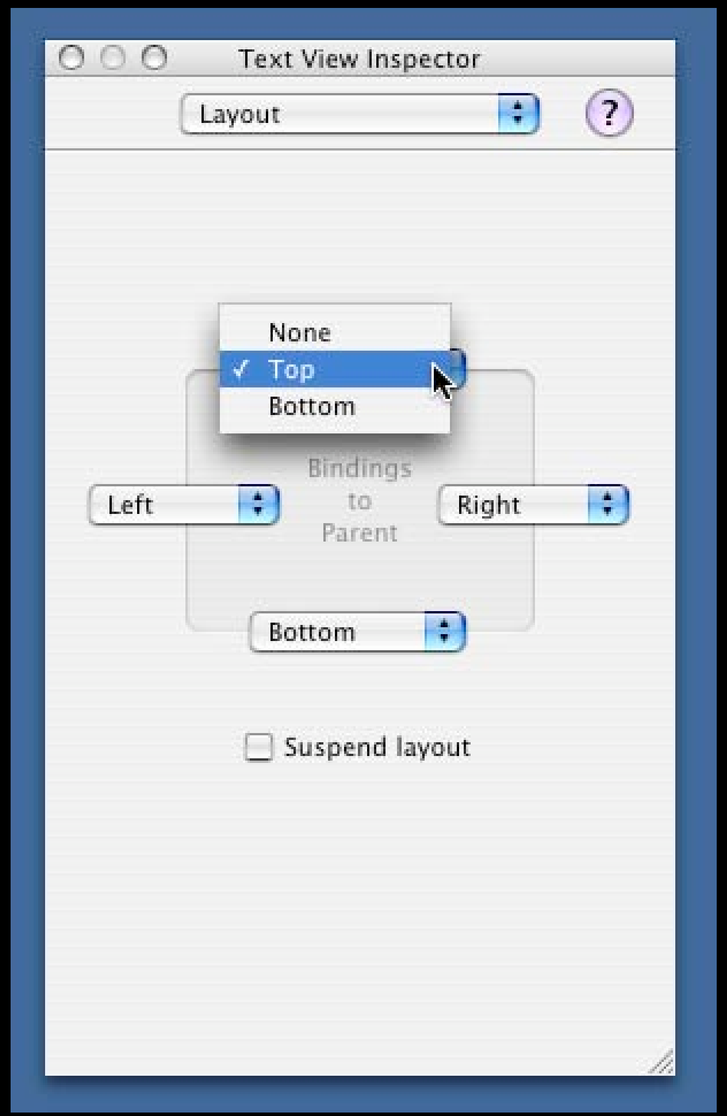
Getting Started With HITextView

Adding resize support

```
HIViewRef textView;
HILayoutInfo layoutInfo;

// ...

layoutInfo.binding.left.kind = kHILayoutBindLeft;
layoutInfo.binding.left.toView = NULL;
layoutInfo.binding.left.offset = 0.0;

// repeat for top, bottom, and right edges

status = HIViewSetLayoutInfo(textView, &layoutInfo);
```

Getting Started With HITextView

Adding resize support (*via Interface Builder*)



Demo

Adding an HITextView to your application

Aaron Haney
Text Engineer

http://developer.apple.com/wwdc2005/

Adding Scrolling Support

- Embed the HITextView in an HIScrollView
- Options controlling scrolling behavior are passed to HIScrollView

```
HIViewRef scrollView, textView;
OptionBits options;

// ...

// Create the ScrollView
options = kHIScrollViewOptionsVertScroll;
status = HIScrollViewCreate(options, &scrollView);

// Embed the TextView in it
status = HIViewAddSubview(scrollView, textView);
```

Adding Scrolling Support

- Embed the HITextView in an HIScrollView
- Options controlling scrolling behavior are passed to HIScrollView

```
HIViewRef scrollView, textView;
OptionBits options;

// ...

// Create the ScrollView
options = kHIScrollViewOptionsVertScroll;
status = HIScrollViewCreate(options, &scrollView);

// Embed the TextView in it
status = HIViewAddSubview(scrollView, textView);
```

Adding Scrolling Support

- Embed the HITextView in an HIScrollView
- Options controlling scrolling behavior are passed to HIScrollView

```
HIViewRef scrollView, textView;
OptionBits options;

// ...

// Create the ScrollView
options = kHIScrollViewOptionsVertScroll;
status = HIScrollViewCreate(options, &scrollView);

// Embed the TextView in it
status = HIViewAddSubview(scrollView, textView);
```

Adding Scrolling Support

- Embed the HITextView in an HIScrollView
- Options controlling scrolling behavior are passed to HIScrollView

```
HIViewRef scrollView, textView;
OptionBits options;

// ...

// Create the ScrollView
options = kHIScrollViewOptionsVertScroll;
status = HIScrollViewCreate(options, &scrollView);

// Embed the TextView in it
status = HIViewAddSubview(scrollView, textView);
```

Adding Scrolling Support

- Embed the HITextView in an HIScrollView
- Options controlling scrolling behavior are passed to HIScrollView

```
HIViewRef scrollView, textView;
OptionBits options;

// ...

// Create the ScrollView
options = kHIScrollViewOptionsVertScroll;
status = HIScrollViewCreate(options, &scrollView);

// Embed the TextView in it
status = HIViewAddSubview(scrollView, textView);
```

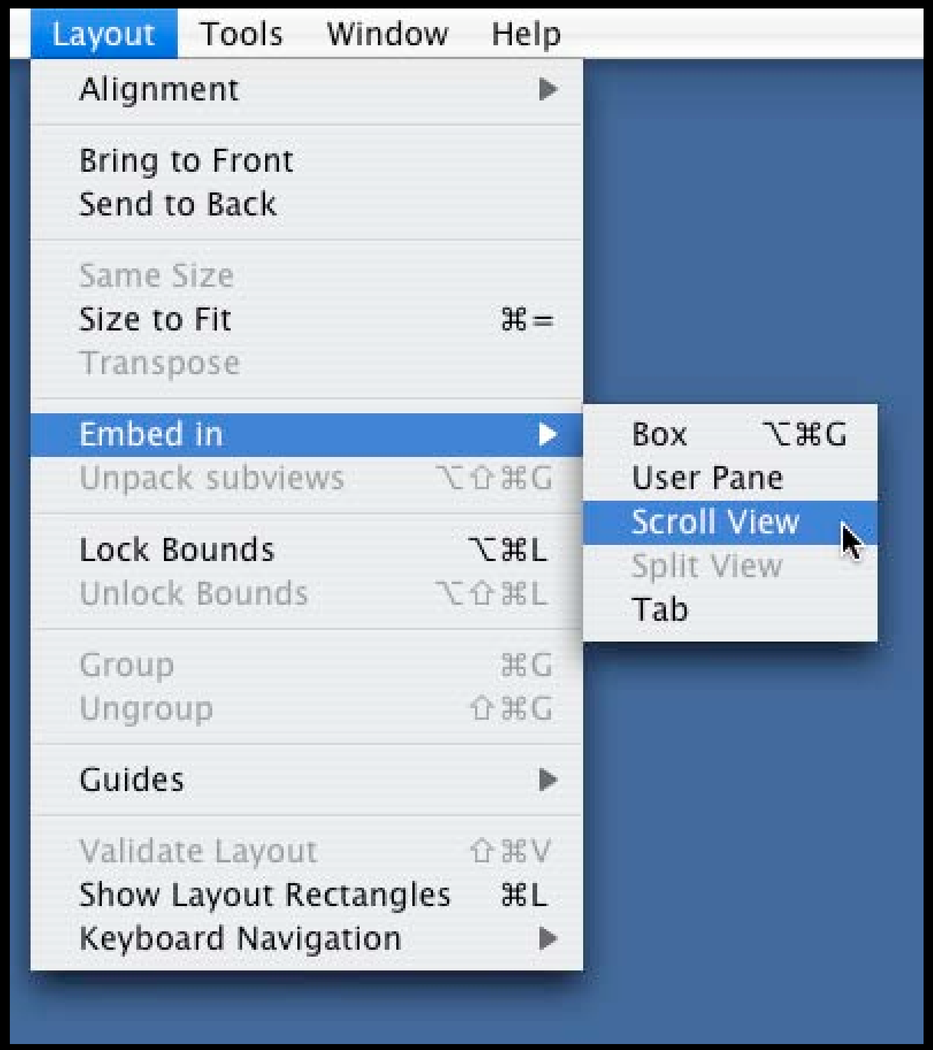
Adding Scrolling Support

- Create an HIScrollView and embed the HITextView in it
- Options controlling scrolling behavior are passed to HIScrollView

```
HIViewRef scrollView, textView;
OptionBits options;

// ...

// Create the ScrollView
options = kHIScrollViewOptionsVertScroll;
status = HIScrollViewCreate(options, &scrollView);

// Embed the TextView in it
status = HIViewAddSubview(scrollView, textView);

// That's all!
```

Adding Scrolling Support

Can also be done in Interface Builder



Demo

Adding scrolling support to your HITextView

http://developer.apple.com/wwdc2005/

Adding Font Panel Support

- Create a UI element (usually a menu) to send the command event `kHICommandShowHideFontPanel ('shfp')`
- Set the HITextView's TXNObject to handle font panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;


txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportFontCommandProcessing;
options |= kTXNSupportFontCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Adding Font Panel Support

- Create a UI element (usually a menu) to send the command event `kHICommandShowHideFontPanel ('shfp')`
- Set the HITextView's TXNObject to handle font panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;


txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportFontCommandProcessing;
options |= kTXNSupportFontCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Adding Font Panel Support

- Create a UI element (usually a menu) to send the command event `kHICommandShowHideFontPanel ('shfp')`
- Set the HITextView's TXNObject to handle font panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;


txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportFontCommandProcessing;
options |= kTXNSupportFontCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Adding Font Panel Support

- Create a UI element (usually a menu) to send the command event `kHICommandShowHideFontPanel ('shfp')`
- Set the HITextView's TXNObject to handle font panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;


txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportFontCommandProcessing;
options |= kTXNSupportFontCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Adding Font Panel Support

- Create a UI element (usually a menu) to send the command event `kHICommandShowHideFontPanel ('shfp')`
- Set the HITextView's TXNObject to handle font panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;


txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportFontCommandProcessing;
options |= kTXNSupportFontCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Adding Font Panel Support

- Create a UI element (usually a menu) to send the command event `kHICommandShowHideFontPanel ('shfp')`
- Set the HITextView's TXNObject to handle font panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;


txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportFontCommandProcessing;
options |= kTXNSupportFontCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Demo

Adding font panel support to your HITextView

http://developer.apple.com/wwdc2005/

Adding Spelling Panel Support

- Create UI elements to send the command events `kHICommandCheckSpelling`, `kHICommandShowSpellingPanel`, and `kHICommandCheckSpellingAsYouType`
- Set the HITextView's TXNObject to handle spelling panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;

txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportSpellCheckCommandProcessing;
options |= kTXNSupportSpellCheckCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Adding Spelling Panel Support

- Create UI elements to send the command events `kHICommandCheckSpelling`, `kHICommandShowSpellingPanel`, and `kHICommandCheckSpellingAsYouType`
- Set the HITextView's TXNObject to handle spelling panel events:

```
HIViewRef textView;
TXNCommandEventSupportOptions options;
TXNObject txnObj;

txnObj = HITextViewGetTXNObject(textView);

status = TXNGetCommandEventSupport(txnObj, &options);

options |= kTXNSupportSpellCheckCommandProcessing;
options |= kTXNSupportSpellCheckCommandUpdating;

status = TXNSetCommandEventSupport(txnObj, options);
```

Demo

Adding spelling panel support to your HITextView

http://developer.apple.com/wwdc2005/

Adding Transparency

- To add transparency to an HITextView, use the function `HITextViewSetBackgroundColor` with a non-opaque color

```
HIViewRef textView;
CGColorRef prevColor, newColor;
float alpha = 0.50;

// ...

status = HITextViewCopyBackgroundColor(textView, &prevColor);

newColor = CGColorCreateCopyWithAlpha(prevColor, alpha);
CGColorRelease(prevColor);

status = HITextViewSetBackgroundColor(textView, newColor);
CGColorRelease(newColor);
```

Adding Transparency

- To add transparency to an HITextView, use the function `HITextViewSetBackgroundColor` with a non-opaque color

```
HIViewRef textView;
CGColorRef prevColor, newColor;
float alpha = 0.50;

// ...

status = HITextViewCopyBackgroundColor(textView, &prevColor);

newColor = CGColorCreateCopyWithAlpha(prevColor, alpha);
CGColorRelease(prevColor);

status = HITextViewSetBackgroundColor(textView, newColor);
CGColorRelease(newColor);
```

Adding Transparency

- To add transparency to an HITextView, use the function `HITextViewSetBackgroundColor` with a non-opaque color

```
HIViewRef textView;
CGColorRef prevColor, newColor;
float alpha = 0.50;

// ...

status = HITextViewCopyBackgroundColor(textView, &prevColor);

newColor = CGColorCreateCopyWithAlpha(prevColor, alpha);
CGColorRelease(prevColor);

status = HITextViewSetBackgroundColor(textView, newColor);
CGColorRelease(newColor);
```

Adding Transparency

- To add transparency to an HITextView, use the function `HITextViewSetBackgroundColor` with a non-opaque color

```
HIViewRef textView;
CGColorRef prevColor, newColor;
float alpha = 0.50;

// ...

status = HITextViewCopyBackgroundColor(textView, &prevColor);

newColor = CGColorCreateCopyWithAlpha(prevColor, alpha);
CGColorRelease(prevColor);

status = HITextViewSetBackgroundColor(textView, newColor);
CGColorRelease(newColor);
```

Adding Transparency

- To add transparency to an HITextView, use the function `HITextViewSetBackgroundColor` with a non-opaque color

```
HIViewRef textView;
CGColorRef prevColor, newColor;
float alpha = 0.50;

// ...

status = HITextViewCopyBackgroundColor(textView, &prevColor);

newColor = CGColorCreateCopyWithAlpha(prevColor, alpha);
CGColorRelease(prevColor);

status = HITextViewSetBackgroundColor(textView, newColor);
CGColorRelease(newColor);
```

Adding Transparency

- To add transparency to an HITextView, use the function `HITextViewSetBackgroundColor` with a non-opaque color

```
HIViewRef textView;
CGColorRef prevColor, newColor;
float alpha = 0.50;

// ...

status = HITextViewCopyBackgroundColor(textView, &prevColor);

newColor = CGColorCreateCopyWithAlpha(prevColor, alpha);
CGColorRelease(prevColor);

status = HITextViewSetBackgroundColor(textView, newColor);
CGColorRelease(newColor);
```

Demo

Adding transparency to your HITextView

http://developer.apple.com/wwdc2005/

Responding to File Menu Commands

- New Tiger APIs for reading and writing
  - TXNReadFromCFURL
  - TXNWriteRangeToCFURL

Reading From a CFURL

- To read from a CFURL, with file type auto-detection, use the following call:

```
HIViewRef textView;
CFURLRef fileURL;

//...

TXNReadFromCFURL(HITextViewGetTXNObject(textView),
                 kTXNStartOffset,
                 kTXNEndOffset,
                 NULL,    // DataOptions  (CFDictionary)
                 fileURL,
                 NULL);   // DocAttributes (CFDictionary)
```

Reading From a CFURL

- To read from a CFURL, with file type auto-detection, use the following call:

```
HIViewRef textView;
CFURLRef fileURL;

//...

TXNReadFromCFURL(HITextViewGetTXNObject(textView),
                 kTXNStartOffset,
                 kTXNEndOffset,
                 NULL,    // DataOptions  (CFDictionary)
                 fileURL,
                 NULL);   // DocAttributes (CFDictionary)
```

Reading From a CFURL

- To read from a CFURL, with file type auto-detection, use the following call:

```
HIViewRef textView;
CFURLRef fileURL;

//...

TXNReadFromCFURL(HITextViewGetTXNObject(textView),
                 kTXNStartOffset,
                 kTXNEndOffset,
                 NULL,    // DataOptions  (CFDictionary)
                 fileURL,
                 NULL);   // DocAttributes (CFDictionary)
```

Reading From a CFURL

- To read from a CFURL, with file type auto-detection, use the following call:

```
HIViewRef textView;
CFURLRef fileURL;

//...

TXNReadFromCFURL(HITextViewGetTXNObject(textView),
                 kTXNStartOffset,
                 kTXNEndOffset,
                 NULL,    // DataOptions  (CFDictionary)
                 fileURL,
                 NULL);   // DocAttributes (CFDictionary)
```

Reading From a CFURL

- To read from a CFURL, with file type auto-detection, use the following call:

```
HIViewRef textView;
CFURLRef fileURL;

//...

TXNReadFromCFURL(HITextViewGetTXNObject(textView),
                 kTXNStartOffset,
                 kTXNEndOffset,
                 NULL,    // DataOptions  (CFDictionary)
                 fileURL,
                 NULL);   // DocAttributes (CFDictionary)
```

Reading From a CFURL

- To read from a CFURL, with file type auto-detection, use the following call:

```
HIViewRef textView;
CFURLRef fileURL;

//...

TXNReadFromCFURL(HITextViewGetTXNObject(textView),
                 kTXNStartOffset,
                 kTXNEndOffset,
                 NULL,    // DataOptions  (CFDictionary)
                 fileURL,
                 NULL);   // DocAttributes (CFDictionary)
```

Writing to a CFURL

- Similar to TXNReadFromCFURL
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,  // DataOptions   (CFDictionary)
                     NULL,         // DocAttributes (CFDictionary)
                     fileURL);
```

Writing to a CFURL

- Similar to TXNReadFromCFURL
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,  // DataOptions   (CFDictionary)
                     NULL,         // DocAttributes (CFDictionary)
                     fileURL);
```

Writing to a CFURL

- Similar to TXNReadFromCFURL
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,   // DataOptions   (CFDictionary)
                     NULL,          // DocAttributes (CFDictionary)
                     fileURL);
```

Writing to a CFURL

- Similar to `TXNReadFromCFURL`
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,   // DataOptions   (CFDictionary)
                     NULL,          // DocAttributes (CFDictionary)
                     fileURL);
```

Writing to a CFURL

- Similar to `TXNReadFromCFURL`
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,   // DataOptions   (CFDictionary)
                     NULL,          // DocAttributes (CFDictionary)
                     fileURL);
```

Writing to a CFURL

- Similar to TXNReadFromCFURL
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,  // DataOptions   (CFDictionary)
                     NULL,         // DocAttributes (CFDictionary)
                     fileURL);
```

Writing to a CFURL

- Similar to TXNReadFromCFURL
- A CFDictionary supplies information about file type

```
HIViewRef textView;
CFURLRef fileURL;

// ...

TXNWriteRangeToCFURL(HITextViewGetTXNObject(textView),
                     kTXNStartOffset,
                     kTXNEndOffset,
                     dataOptions,  // DataOptions   (CFDictionary)
                     NULL,         // DocAttributes (CFDictionary)
                     fileURL);
```

DataOptions Dictionary

Contains information about document type

Keys	Values
kTXNDataOptionDocumentTypeKey	kTXNRTFDocumentType kTXNPlainTextDocumentType kTXNMLTEDocumentType kTXNQuickTimeDocumentType *(loading only)*
kTXNDataOptionCharacterEncodingKey	kCFStringEncoding* *(stored as CFNumber)*

Creating a DataOptions CFDictionary

- Supplies information about the file type to read or write
- Only one key/value pair for most document types

```
CFDictionaryRef dataOptions;

dataOptions =
CFDictionaryCreate(NULL,
(const void **)&kTXNDataOptionDocumentTypeKey,
(const void **)&kTXNRTFDocumentType,
1,
NULL,
NULL);
```

Creating a DataOptions CFDictionary

- Supplies information about the file type to read or write
- Only one key/value pair for most document types

```
CFDictionaryRef dataOptions;

dataOptions =
CFDictionaryCreate(NULL,
(const void **)&kTXNDataOptionDocumentTypeKey,
(const void **)&kTXNRTFDocumentType,
1,
NULL,
NULL);
```

Creating a DataOptions CFDictionary

- Supplies information about the file type to read or write
- Only one key/value pair for most document types

```
CFDictionaryRef dataOptions;

dataOptions =
CFDictionaryCreate(NULL,
(const void **)&kTXNDataOptionDocumentTypeKey,
(const void **)&kTXNRTFDocumentType,
1,
NULL,
NULL);
```

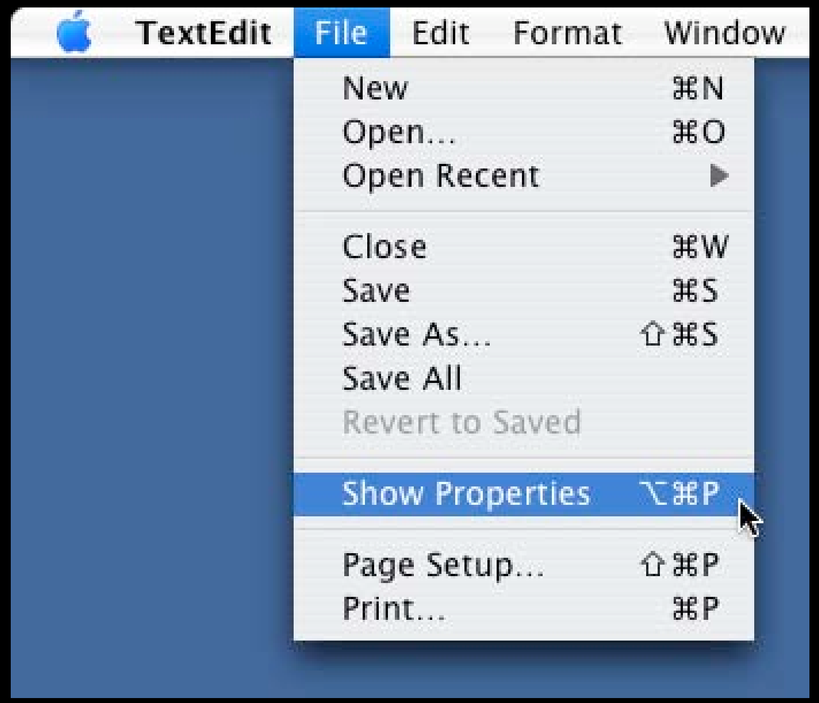
Creating a DataOptions CFDictionary

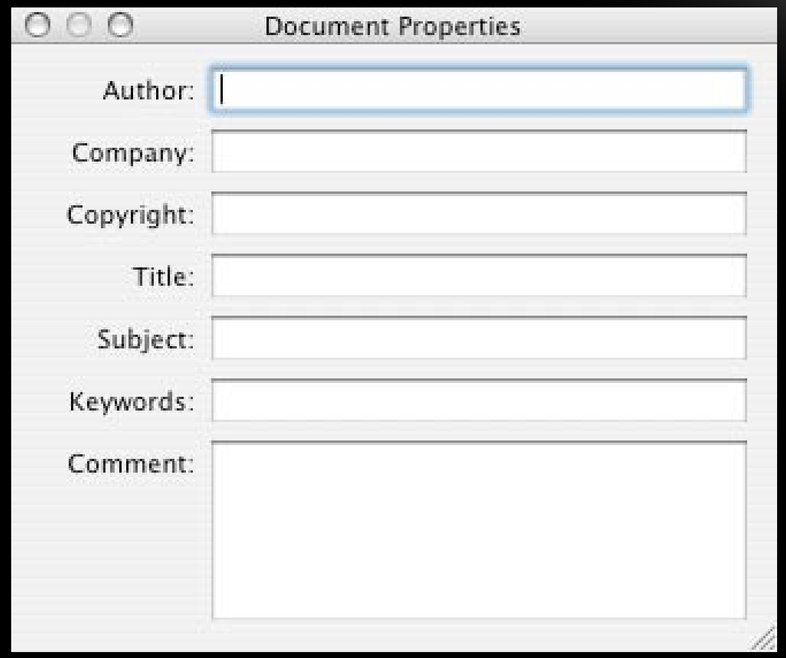
- Supplies information about the file type to read or write
- Only one key/value pair for most document types

```
CFDictionaryRef dataOptions;

dataOptions =
CFDictionaryCreate(NULL,
(const void **)&kTXNDataOptionDocumentTypeKey,
(const void **)&kTXNRTFDocumentType,
1,
NULL,
NULL);
```

Document Attributes CFDictionary

- Provides access to document properties for Rich Text Format (RTF) and MLTE format documents





Document Attributes CFDictionary

- Keys are predefined in MacTextEditor.h
- All values are CF types; most are CFStringRef

Key	Value
kTXNDocumentAttributeTitleKey	CFString
kTXNDocumentAttributeCompanyNameKey	CFString
kTXNDocumentAttributeSubjectKey	CFString
kTXNDocumentAttributeAuthorKey	CFString
kTXNDocumentAttributeKeywordsKey	CFArray of CFStrings
kTXNDocumentAttributeCommentKey	CFString
kTXNDocumentAttributeEditorKey	CFString
kTXNDocumentAttributeCreationTimeKey	CFAbsoluteTime
kTXNDocumentAttributeModificationTimeKey	CFAbsoluteTime
kTXNDocumentAttributeCopyrightKey	CFString

Demo

Responding to file menu commands in HITextView

http://developer.apple.com/wwdc2005/

Summary

- HITextView rocks!
- It’s easy to
  - Create and embed
  - Add font and spelling panel support
  - Read/write files

More Information

http://developer.apple.com/wwdc2005/

Related Sessions

Carbon and HIToolbox Feedback Forum	Session 131 Thursday 10:30AM
Text, Fonts, and International Technologies Feedback Forum	Session 140 Friday 9:00AM
Take Your Application Global	Session 141 Friday 10:30AM

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Who to Contact

Xavier Legros
Worldwide Developer Relations
xavier@apple.com

Q&A

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference